Lab 11
CNN

Sung Kim <hunkim+ml@gmail.com>
http://hunkim.github.io/ml/

TensorFlow-Tutorials

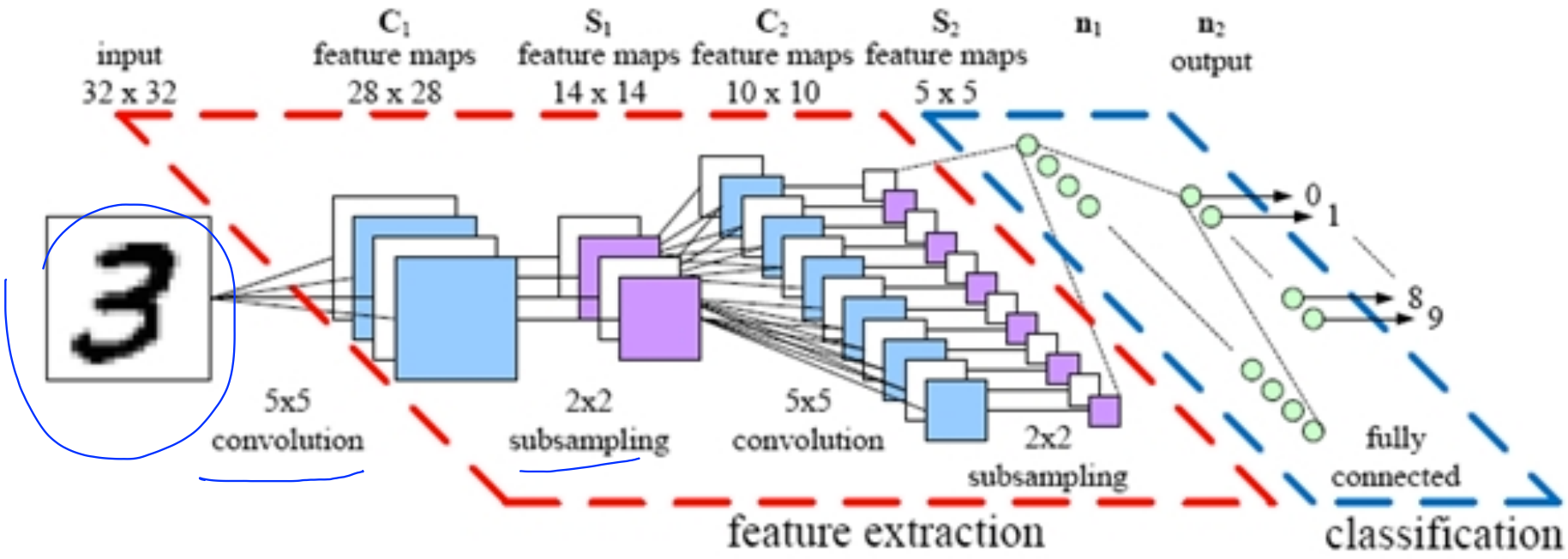
Introduction to deep learning based on Google's TensorFlow framework. These tutorials are direct ports of Newmu's Theano Tutorials

Topics

- Simple Multiplication
- Linear Regression
- Logistic Regression
- Feedforward Neural Network (Multilayer Perceptron)
- Deep Feedforward Neural Network (Multilayer Perceptron with 2 Hidden Layers O.o)
- Convolutional Neural Network
- Denoising Autoencoder
- LSTM

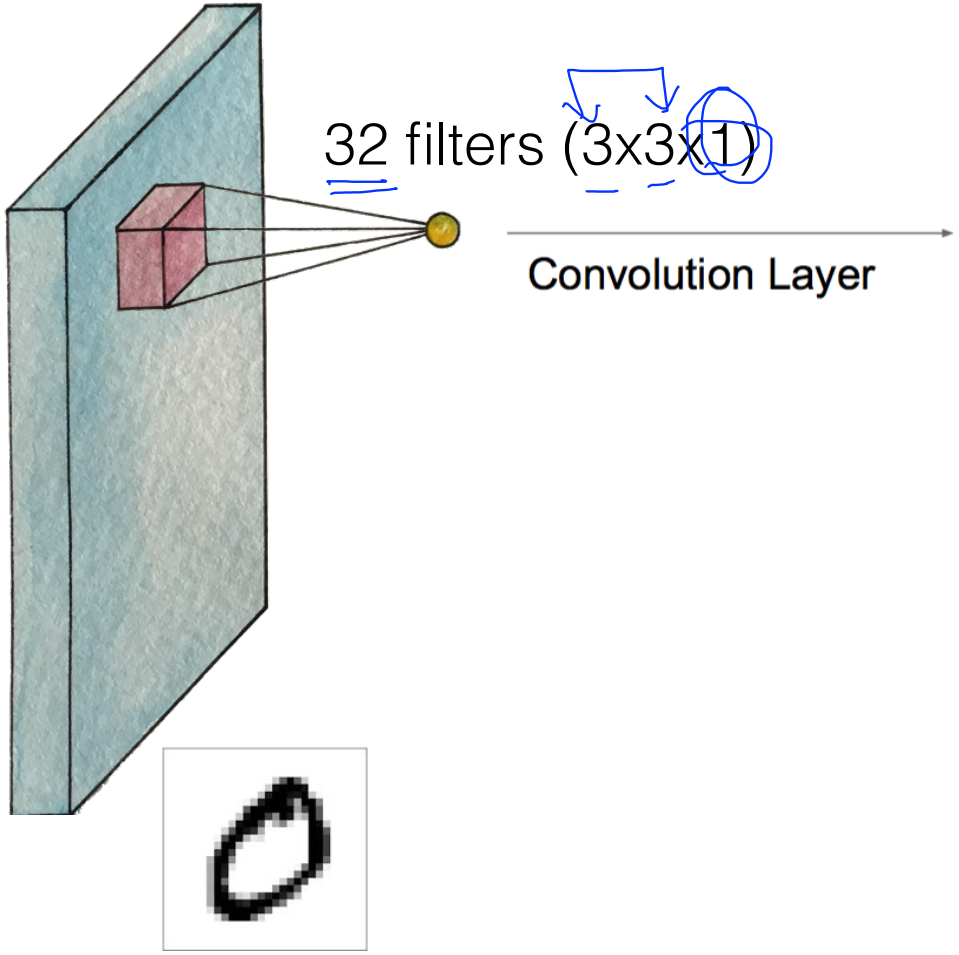
https://github.com/nlintz/TensorFlow-Tutorials

CNN



http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

Convolutional layers



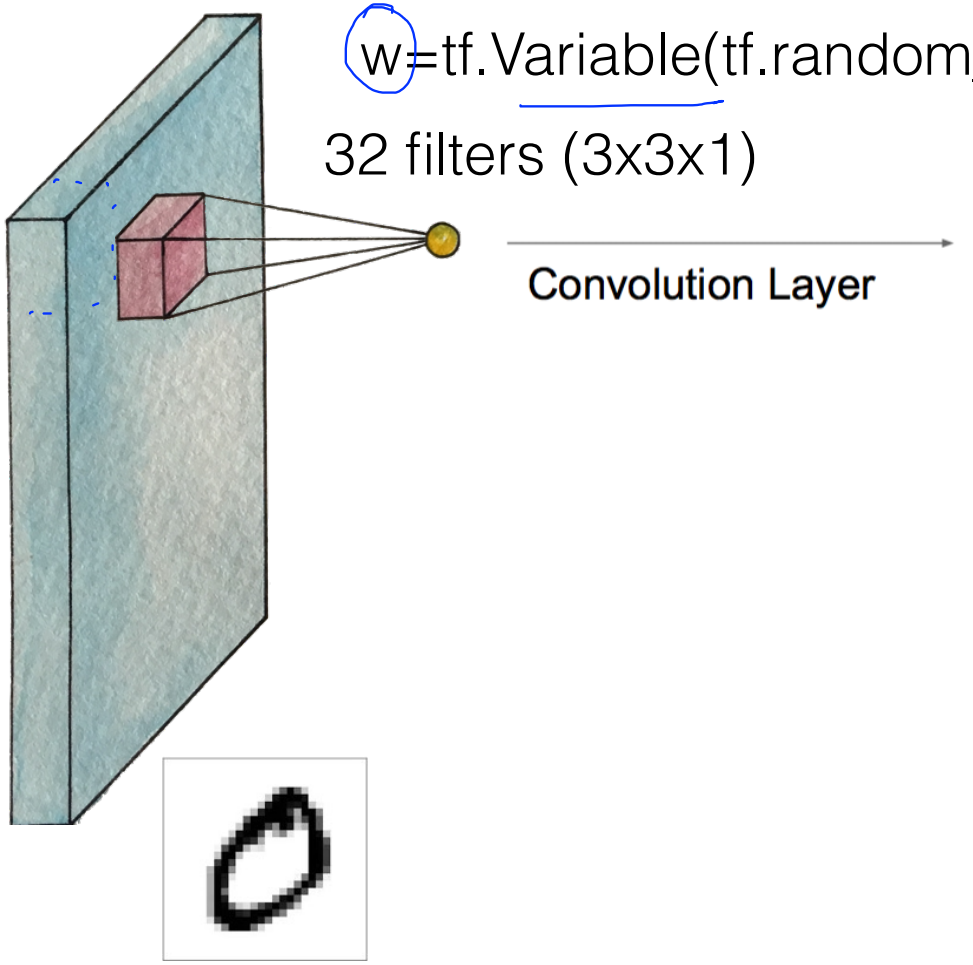
MNIST 28x28x1 image

Convolutional layers

w=tf.Variable(tf.random_normal([3,3,1,32], stddev=0.01))

output layers

32 filters (3x3x1)

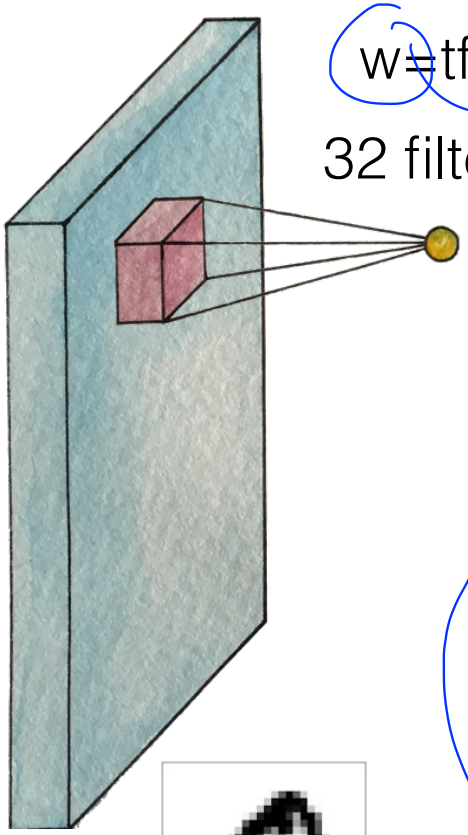


28x28x1 image

Convolutional layers

w=tf.Variable(tf.random_normal([3,3,1,32], stddev=0.01))

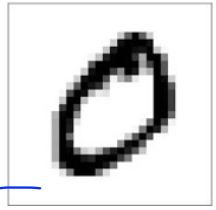
32 filters (3x3x1)



Convolution Layer

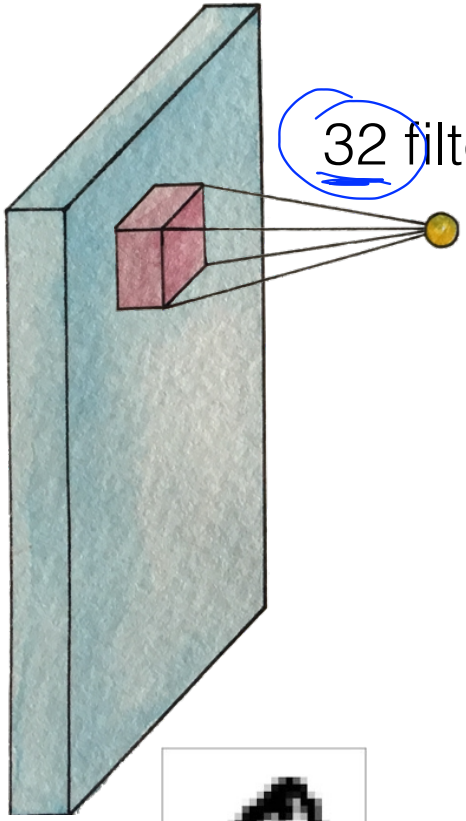
```
tf.nn.conv2d(X, w,
    strides=[1, 1, 1, 1], padding='SAME')
```

```
strides = [1, stride, stride, 1]
```



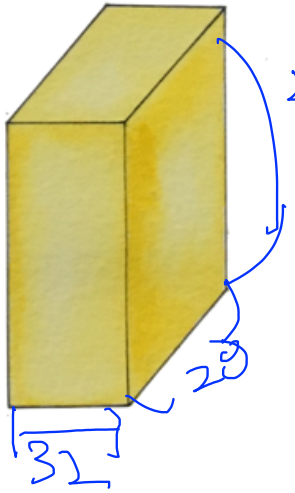
28x28x1 image

Convolutional layers



32 filters (3x3x1)

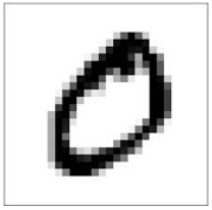
Convolution Layer



activation maps

(?, ?, 32)

```
tf.nn.conv2d(X, w,
   strides=[1, 1, 1, 1], padding='SAME')
```

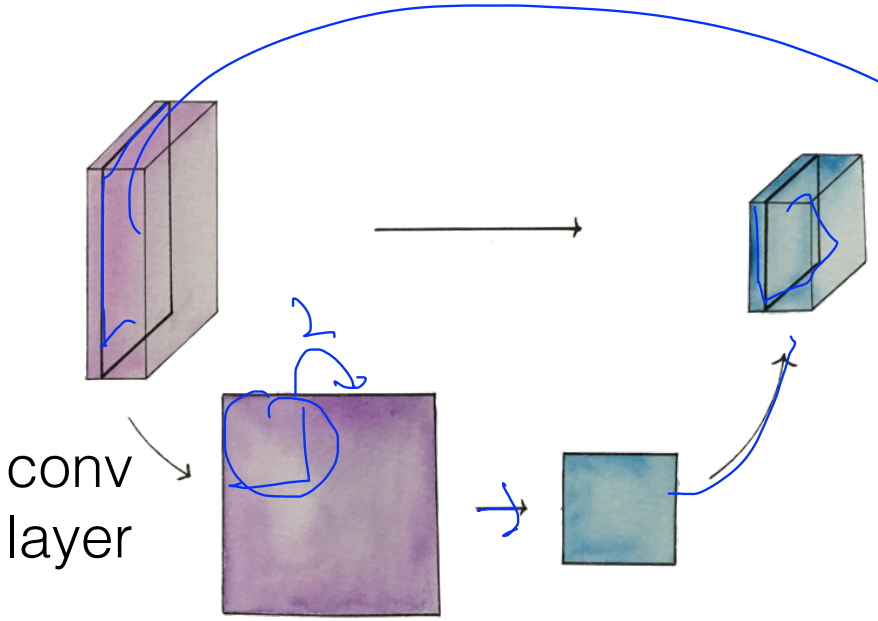


28x28x1 image

ReLU

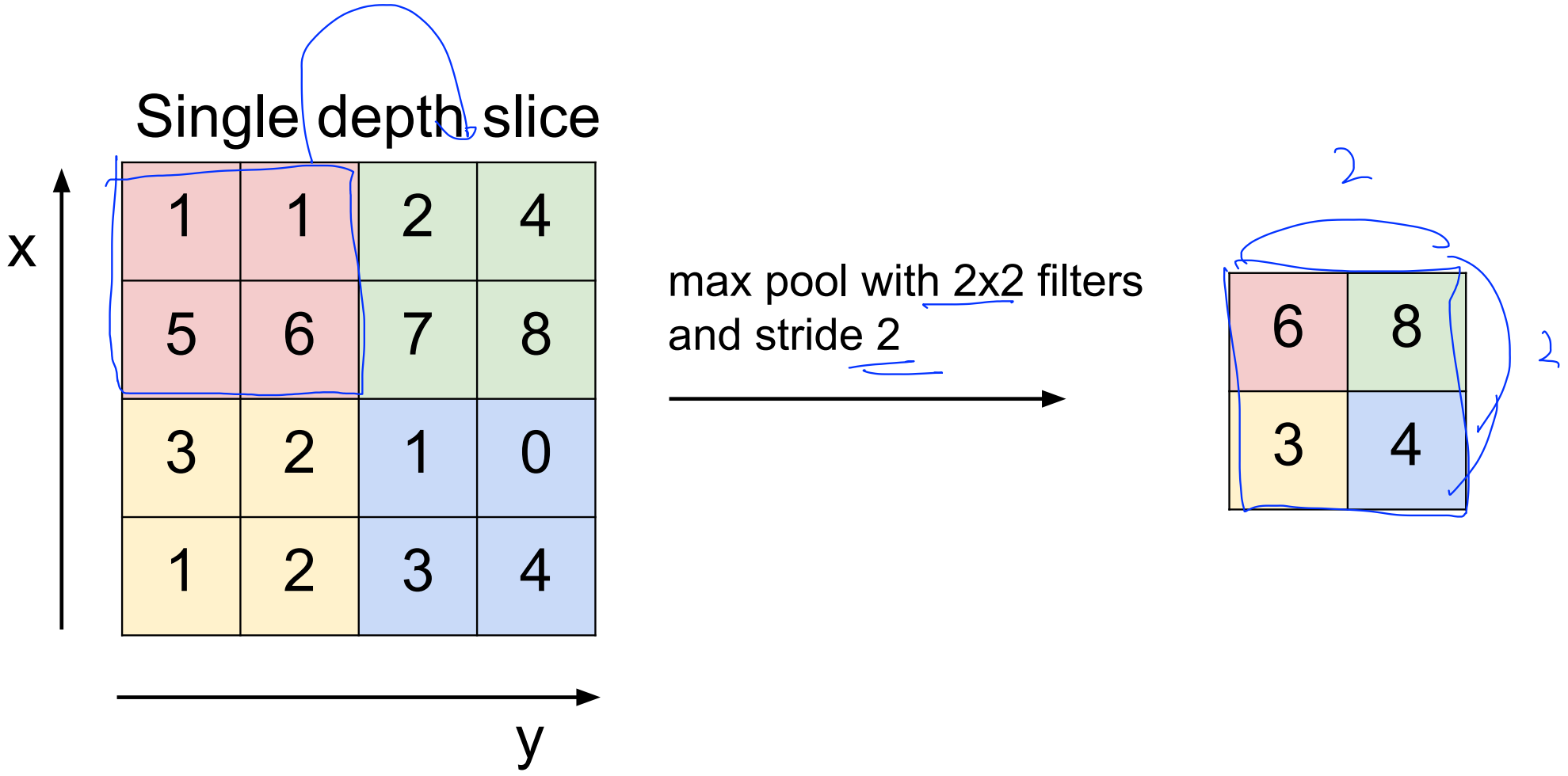
```
l = tf.nn.conv2d(X, w, [1, 1, 1, 1], 'SAME')

l1a = tf.nn.relu(tf.nn.conv2d(X, w, [1, 1, 1, 1], 'SAME'))
```

Pooling layer (sampling)

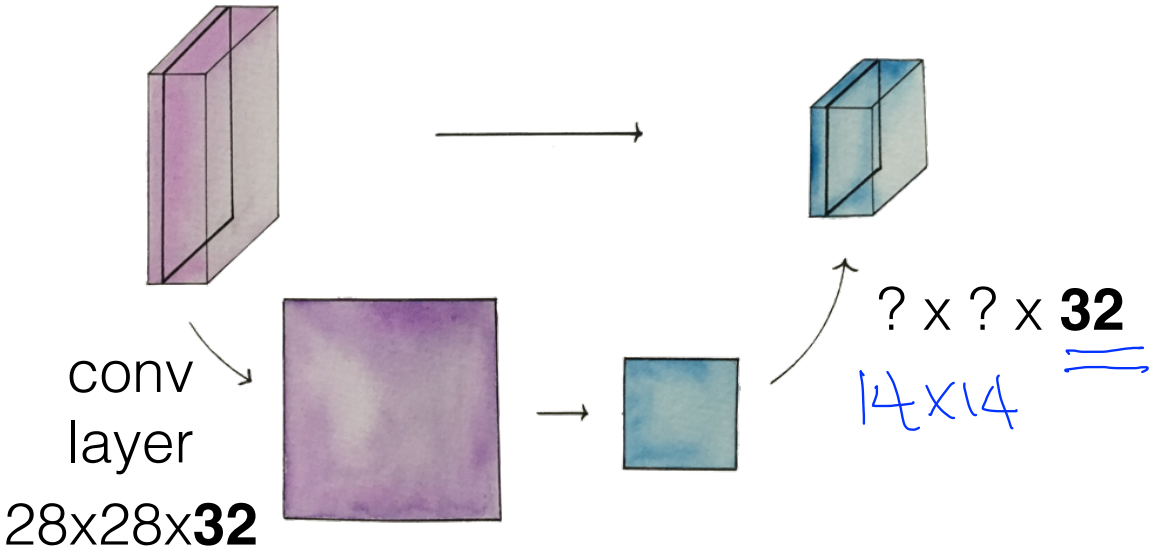



```
l1 = tf.nn.max_pool(c1, ksize=[1, 2, 2, 1],
      strides=[1, 2, 2, 1], padding='SAME')
```

MAX POOLING

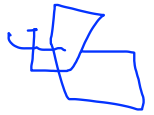

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Pooling layer (sampling)




```
l1 = tf.nn.max_pool(c1, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
```

Shape not sure? Print tensor



```
l1a = tf.nn.relu(tf.nn.conv2d(X, w, [1, 1, 1, 1], 'SAME'))
print l1a
```

Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)

```
l1 = tf.nn.max_pool(l1a, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
print l1
```

Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)

```
X = trX.reshape(-1, 28, 28, 1)
```

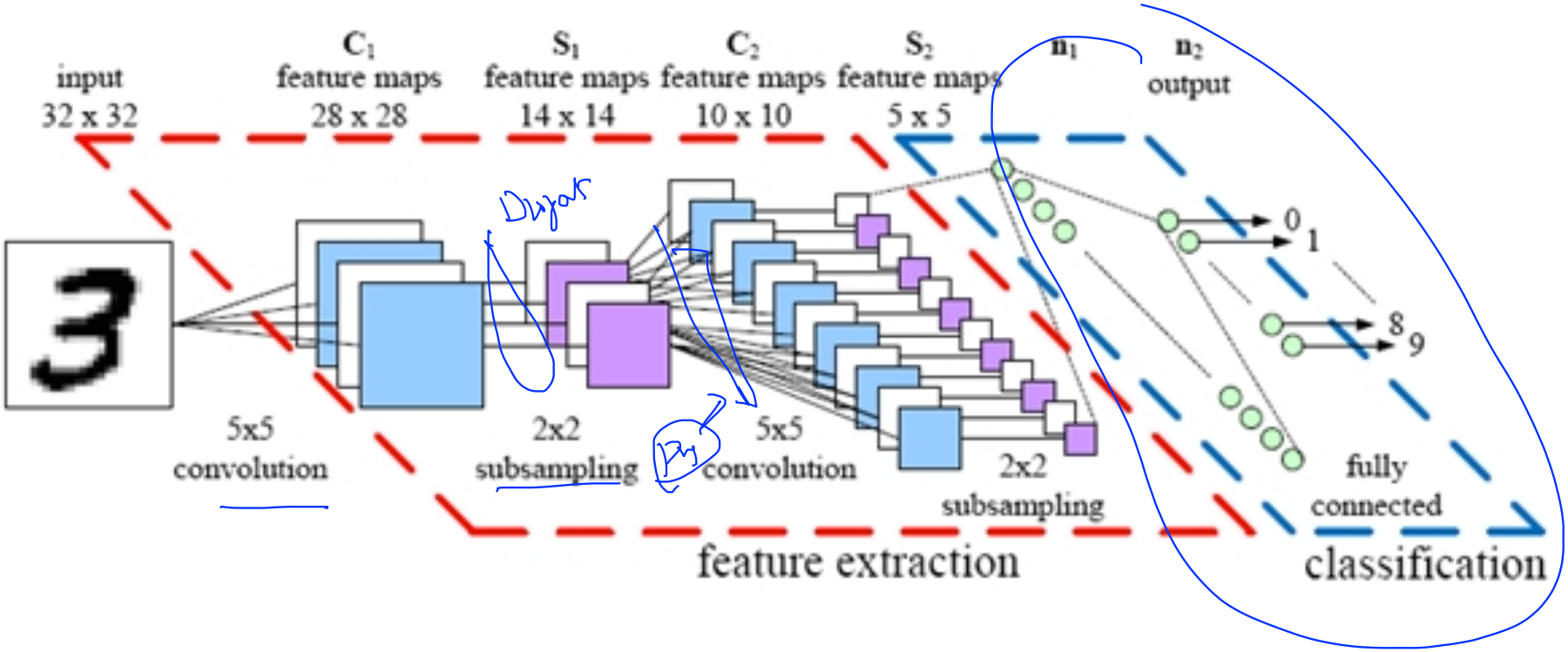
```
w  = init_weights([3, 3, 1, 32])       # 3x3x1 conv, 32 outputs
w2 = init_weights([3, 3, 32, 64])      # 3x3x32 conv, 64 outputs
w3 = init_weights([3, 3, 64, 128])     # 3x3x32 conv, 128 outputs

l1a = tf.nn.relu(tf.nn.conv2d(X, w,                     # l1a shape=(?, 28, 28, 32)
                              strides=[1, 1, 1, 1], padding='SAME'))
l1 = tf.nn.max_pool(l1a, ksize=[1, 2, 2, 1],            # l1 shape=(?, 14, 14, 32)
                    strides=[1, 2, 2, 1], padding='SAME')

l2a = tf.nn.relu(tf.nn.conv2d(l1, w2,                   # l2a shape=(?, 14, 14, 64)
                              strides=[1, 1, 1, 1], padding='SAME'))
l2 = tf.nn.max_pool(l2a, ksize=[1, 2, 2, 1],            # l2 shape=(?, 7, 7, 64)
                    strides=[1, 2, 2, 1], padding='SAME')

l3a = tf.nn.relu(tf.nn.conv2d(l2, w3,                   # l3a shape=(?, 7, 7, 128)
                              strides=[1, 1, 1, 1], padding='SAME'))
l3 = tf.nn.max_pool(l3a, ksize=[1, 2, 2, 1],            # l3 shape=(?, 4, 4, 128)
                    strides=[1, 2, 2, 1], padding='SAME')
```

dropout

X = trX.reshape(-1, 28, 28, 1)

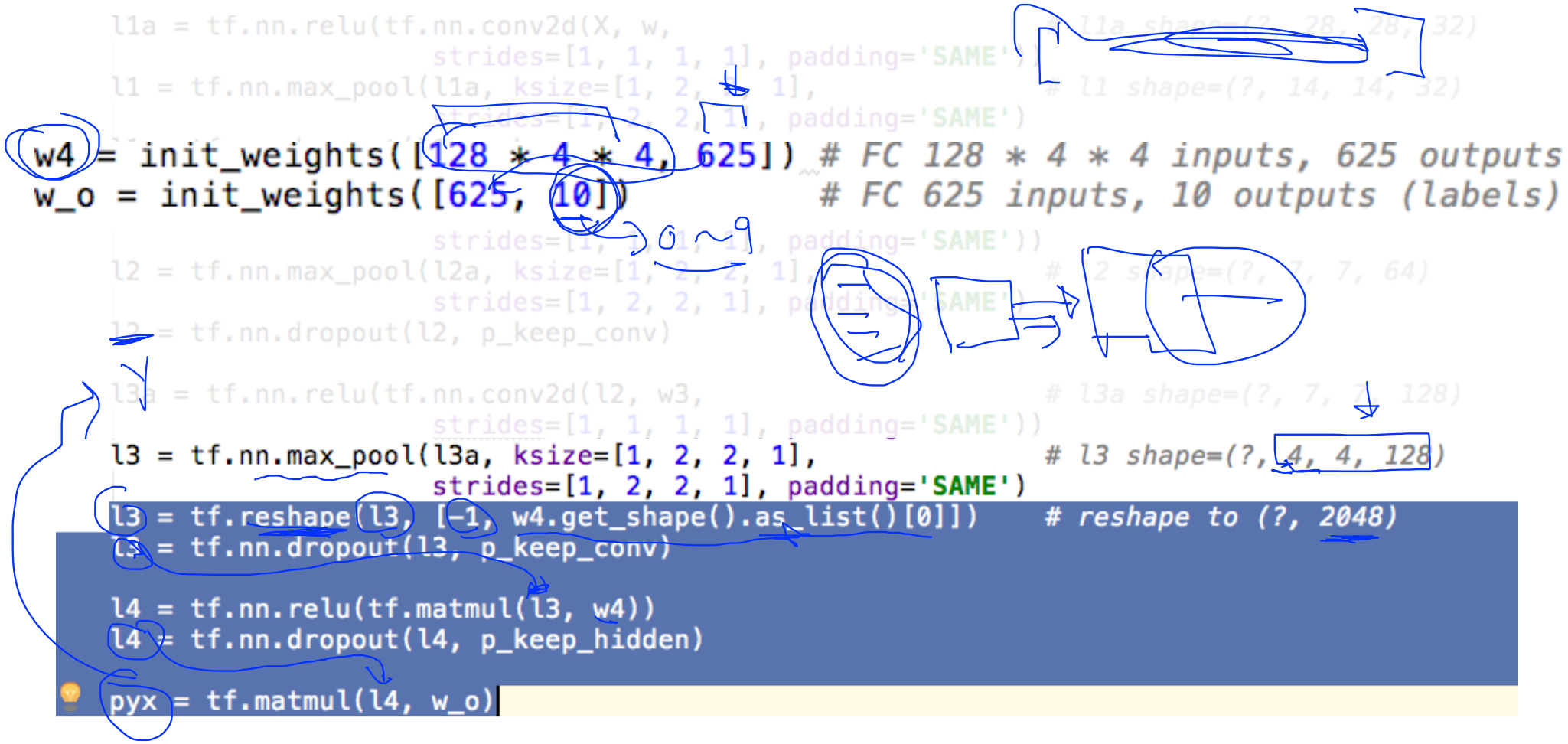
```
l1a = tf.nn.relu(tf.nn.conv2d(X, w,                     # l1a shape=(?, 28, 28, 32)
                    strides=[1, 1, 1, 1], padding='SAME'))
l1 = tf.nn.max_pool(l1a, ksize=[1, 2, 2, 1],            # l1 shape=(?, 14, 14, 32)
                    strides=[1, 2, 2, 1], padding='SAME')
l1 = tf.nn.dropout(l1, p_keep_conv)

l2a = tf.nn.relu(tf.nn.conv2d(l1, w2,                   # l2a shape=(?, 14, 14, 64)
                    strides=[1, 1, 1, 1], padding='SAME'))
l2 = tf.nn.max_pool(l2a, ksize=[1, 2, 2, 1],            # l2 shape=(?, 7, 7, 64)
                    strides=[1, 2, 2, 1], padding='SAME')
l2 = tf.nn.dropout(l2, p_keep_conv)

l3a = tf.nn.relu(tf.nn.conv2d(l2, w3,                   # l3a shape=(?, 7, 7, 128)
                    strides=[1, 1, 1, 1], padding='SAME'))
l3 = tf.nn.max_pool(l3a, ksize=[1, 2, 2, 1],            # l3 shape=(?, 4, 4, 128)
                    strides=[1, 2, 2, 1], padding='SAME')
```

CNN



http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

Fully connected net



```
w4 = init_weights([128 * 4 * 4, 625]) # FC 128 * 4 * 4 inputs, 625 outputs
w_o = init_weights([625, 10])         # FC 625 inputs, 10 outputs (labels)

l3 = tf.nn.max_pool(l3a, ksize=[1, 2, 2, 1],              # l3 shape=(?, 4, 4, 128)
                    strides=[1, 2, 2, 1], padding='SAME')
l3 = tf.reshape(l3, [-1, w4.get_shape().as_list()[0]])    # reshape to (?, 2048)
l3 = tf.nn.dropout(l3, p_keep_conv)

l4 = tf.nn.relu(tf.matmul(l3, w4))
l4 = tf.nn.dropout(l4, p_keep_hidden)

pyx = tf.matmul(l4, w_o)
```

cost and optimization

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(py_x, Y))
train_op = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost)
predict_op = tf.argmax(py_x, 1)
```

```
tf.train.RMSPropOptimizer.__init__(learning_rate, decay=0.9,
momentum=0.0, epsilon=1e-10, use_locking=False,
name='RMSProp')
```

Construct a new RMSProp optimizer.

Args:

- `learning_rate`: A Tensor or a floating point value. The learning rate.
- `decay`: Discounting factor for the history/coming gradient

https://www.tensorflow.org/versions/r0.8/api_docs/python/train.html#RMSPropOptimizer

Other TF optimizers

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(py_x, Y))
train_op = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost)
predict_op = tf.argmax(py_x, 1)
```

- class tf.train.GradientDescentOptimizer
- class tf.train.AdadeltaOptimizer
- class tf.train.AdagradOptimizer
- class tf.train.MomentumOptimizer
- class tf.train.AdamOptimizer
- class tf.train.FtrlOptimizer
- class tf.train.RMSPropOptimizer

https://www.tensorflow.org/versions/r0.8/api_docs/python/train.html#RMSPropOptimizer

Train and testing

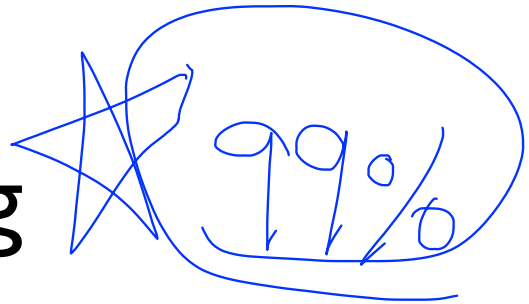
```
# Launch the graph in a session
with tf.Session() as sess:
    # you need to initialize all variables
    tf.initialize_all_variables().run()

    for i in range(100):
        for start, end in zip(range(0, len(trX), 128), range(128, len(trX), 128)):
            sess.run(train_op, feed_dict={X: trX[start:end], Y: trY[start:end],
                                          p_keep_conv: 0.8, p_keep_hidden: 0.5})

        test_indices = np.arange(len(teX)) # Get A Test Batch
        np.random.shuffle(test_indices)
        test_indices = test_indices[0:256]

        print i, np.mean(np.argmax(teY[test_indices], axis=1) ==
                         sess.run(predict_op, feed_dict={X: teX[test_indices],
                                                         Y: teY[test_indices],
                                                         p_keep_conv: 1.0,
                                                         p_keep_hidden: 1.0}))
```

Train and testing

0	128
128	256
256	384
384	512

```
# Launch the graph in a session
with tf.Session() as sess:
    # you need to initialize all variables
    tf.initialize_all_variables().run()

    for i in range(100):
        for start, end in zip(range(0, len(trX), 128), range(128, len(trX), 128)):
            sess.run(train_op, feed_dict={X: trX[start:end], Y: trY[start:end],
                                          p_keep_conv: 0.8, p_keep_hidden: 0.5})

        test_indices = np.arange(len(teX)) # Get A Test Batch
        np.random.shuffle(test_indices)
        test_indices = test_indices[0:256]

        print i, np.mean(np.argmax(teY[test_indices], axis=1) ==
                         sess.run(predict_op, feed_dict={X: teX[test_indices],
                                                         Y: teY[test_indices],
                                                         p_keep_conv: 1.0,
                                                         p_keep_hidden: 1.0}))
```

Train and testing




```
# Launch the graph in a session
with tf.Session() as sess:
    # you need to initialize all variables
    tf.initialize_all_variables().run()

    for i in range(100):
        for start, end in zip(range(0, len(trX), 128), range(128, len(trX
            sess.run(train_op, feed_dict={X: trX[start:end], Y: trY[start
                                          p_keep_conv: 0.8, p_keep_hidder

        test_indices = np.arange(len(teX)) # Get A Test Batch
        np.random.shuffle(test_indices)
        test_indices = test_indices[0:256]

        print i, np.mean(np.argmax(teY[test_indices], axis=1) ==
                         sess.run(predict_op, feed_dict={X: teX[test_indices],
                                                         Y: teY[test_indices],
                                                         p_keep_conv: 1.0,
                                                         p_keep_hidden: 1.0}))
```


```
0 0.98046875
1 0.97265625
2 0.984375
3 0.9765625
4 0.9921875
5 0.9921875
6 0.9921875
```

https://github.com/nlintz/TensorFlow-Tutorials